Named-Entity Recognition using Conditional Random Fields

Md Rafiqul Islam Rabin
PeopleSoft ID: 1797648
mrabin@central.uh.edu

1 Introduction

A named-entity could be anything that is referred to as a proper name such as a person, a title, a product, a location, a group, an organization, and others. Entities are important in question answering, text linking, or sentiment analysis as we like to figure out sentiment towards any particular entities in sentences. The first challenge is to decide whether a word in a sentence is an entity or not. Secondly, the individual type of entity needs to be separated. However, ambiguity makes it quite difficult to process entities in sentences. For example, the smaller case `us` is referred to as a plural personal pronoun, but the capital `US` means the United States. Similarly, we want to match `U.S.A` and `USA` to same entity. Another example, if we say anything like `XYZ`, this could be the name of a person, or location, or any street. Therefore, we need to know the boundary and the type of entity to better recognition.

The goal of this assignment is to learn how to detect and classify those named entities in noisy and unstructured text, which is known as Named-Entity Recognition (`NER`). The named-entity recognition is basically a subtask of information extraction which mainly focuses on transforming raw data into features that better represents the underlying structure of data for a particular target domain. The main task under this assignment is to implement a named entity recognizer using Conditional Random Field (`CRF`).

This report presents the list of features that I have extracted from the raw text for named-entity recognition, and the approach of training `CRF` model on those features to implement the entity recognizer. The model was trained on the training set and evaluated on the development set. The best weighted average F1-score for development set and test set are 27.95% and 27.59%, respectively.

The structure of the paper is organized as follows: This section represents the introduction related to the assignment, Section 2 demonstrates the approach I followed to solve this assignment, which also describes the experimental setup (i.e. dataset, features, model, and other parameters) for solving the assignment. Section 3 provides an analysis of the results and findings. Section 4 discusses some of the challenges that occurred during solving the assignment. Finally, Section 5 concludes the assignment.

2 Methodology

In this section, I will describe the experimental setting and the evaluation approach to solve the assignment. The approach can broadly be divided into four main steps which depict an overall view of the evaluation process: (1) Data preprocessing, (2) Feature extraction, (3) Training the `CRF` model, and (4) Classification task. I will discuss these in the following subsections.

2.1 Data Preprocessing

The dataset given to us for this assignment comes from the `W-NUT` [1] corpus for the training and development set. However, the test set comes from an anonymous social media platform but the test labels were not provided to us. Table 1 summarizes the length of dataset in various aspect.

Dataset	# Posts	# Words	# Named-Entity
Training set	2394	46469	2462
Development set	1000	16261	1128
Test set	4307	42325	-

Table 1: Number of elements in dataset. ('-' denotes the number of true named-entities in test set is unknown.)

The format is the same for all sets - each row in-

[1] https://noisy-text.github.io/

cludes one word with the associated entity-tag separated by a tab, and posts are separated with a blank line from each other. Note that the entity tag is represented as `IOB` encoding, where the prefix **'B-'** means beginning of the entity, **'I-'** means inside of the entity, and **'O'** means this doesn't belong to any named-entity. After the **'B-'** or **'I-'** prefix, it includes seven different types of named-entities (i.e. person, location, group, product, company, title and other).

However, the data could not be directly fitted to the `CRF` model, as the model fits the posts by taking the 'input feature dictionaries' as a list of lists and corresponding 'output tags' as another list of lists. In more detail, the input list (i.e. X) should include all posts, where each sublist represents a unique post, and each element of the sublist is a feature dictionary of a particular word in a post. Similarly, for the corresponding output list (i.e. Y), each element of the sublist is the `IOB` tag of that particular word, which points to the same position of input words. Therefore, to transform the raw data into the `CRF` model as a suitable format, I split the data by blank line to get the list of posts. Then, I iterate over each post and split them by a new line to get a list of words with tags. Next, I create two separate lists of the list for words and tags by splitting each item with tab.

2.2 Feature Extraction

In this section, I extract features from the raw post that improves the quality of the named entity recognizer. The basic features that initially had in my minds are the word itself, length of the word, prefix and suffix, whether the word is lower/upper/title, whether the word contains all or any digit/alphabet/symbols, and whether the word has hyphen/period/comma. After following the class lectures and reading materials given by the instructor (Daniel Jurafsky & James H. Martin, 2019; Jacob Esienstein, 2018), I learn about some more important features that might be helpful to improve the named-entity recognition system.

The followings are the most important features that improve the named-entity recognition system by a large margin in my analysis.

- The `part-of-speech tag` and base-phrase syntactic `chunk tag` better capture both the boundary and the type of entity as a word-by-word sequence in sentence.

- The `gazetteer` has a list of place names with detailed geographical information, so knowing the presence of words can be quite effective.

- The `word shape` feature represents the abstract letter pattern of the word, and another form of it is the `shorter word shape` where the consecutive character types are removed.

- The features from `neighboring words` (previous and next word) also emphasis the context of the current word in sentence.

Features	Model-1	Model-2	Model-3
Word	✓✓	✓✓	✓✓
Length of Word	✓✓	✓✓	×
POS Tag	✓✓	✓✓	✓✓
Chunk Tag	✓✓	✓✓	✓✓
isGazetteer	✓	✓✓	✓✓
Prefix[1-4]	✓	✓	✓
Suffix[1-4]	✓	✓	✓
isUpper	✓	✓✓	✓✓
isLower	×	✓✓	✓✓
isTitle	×	✓✓	✓✓
Word Shape	✓✓	✓✓	✓✓
Short Word Shape	✓✓	✓✓	✓✓
hasHyphen	✓	✓	✓
hasPeriod	×	✓	✓
hasComma	×	✓	✓
hasNumber	×	×	✓
hasCharacter	×	×	✓
hasSymbol	×	×	✓
allNumber	×	✓	✓
allCharacter	×	✓	✓
allSymbol	×	✓	✓
isAlphaNumeric	×	✓	×
isStopword	×	✓✓	✓✓
PorterStemmer	×	✓✓	✓
Lemmatize	×	✓✓	✓

Table 2: The different features set for training the different models. ('✓' denotes the presence of a feature and '×' denotes the absence of a feature in the model. '✓' only captures feature from the current word while '✓✓' captures features from neighboring words (previous and next) alongside current word.)

Note that, I have used `Python NLTK Library`[2] to extract the POS tag, Chunk tag and Gazetteer features from the post. Besides, I also extract PorterStemmer, WordNetLemmatizer, and the presence of stopword to see the influence of the features in named-entity recognition.

[2]`https://www.nltk.org/api/nltk.html`

After that, I have played with a different set of features for different model training that has been summarized in Table 2. Note that the '[1-4]' denotes all length ≤ 4.

Finally, I replace each word of input list by its feature dictionaries for `CRF` input.

2.3 Training the CRF Model

I have implemented my code in `Python3` using the `CRF` model from `Python sklearn-crfsuite Library`[3].

The followings parameter settings are used to build the `CRF` model:

- `algorithm='lbfgs'` : This sets the gradient descent of L-BFGS algorithm.
- `c1=0.1, c2=0.1`: These float coefficients are for L1 and L2 regularization, respectively.
- `max_iterations=100`: The max number of iterations to continue for optimization.
- `all_possible_states=True`: This is used to generate state features that associate with all attributes and labels combination.
- `all_possible_transitions=True`: This is used to generate transition features that associate with all label pairs.

In section 2.2, I explained the structure of feature inputs and labels required for `CRF` model. After data preprocessing and feature extraction, I have trained the `CRF` model on the training set, and tested on the development set to observe the performance of models and tune the parameters based on the observation. In the final step, I run the prediction on the test data.

2.4 Classification Task

The classification task is to predict tags for new unlabeled input. In our context, given an unlabeled post, we have to classify the type of named-entities. To this purpose, first I prepare a feature dictionary for each word of the post and create a list of lists structure for `CRF` model. After that, I perform the prediction using the trained model and the model returns the predicted labels as a list of lists that match with the corresponding position of each word. I use the metrics from `Python seqeval Library`[4] to report the classification score. After getting the prediction on the development set, I pass the predicted labels and true labels to the `seqeval` for classification report. This gives us Precision, Recall, and F1-score for each different type of named-entity separately and also as (micro/macro) average.

3 Analysis of Results

In this section, I will explain my findings from results based on scores, features, and parameters to compare models.

3.1 F1-score vs Accuracy

Table 1 is showing the number of elements in the dataset in various aspects. The '# Named-Entity' column represents the total number of named-entities in a dataset. With respect to the number of words, the number of named-entities is very low. For the training set, only 5.29% words are named-entity, and for the development set, only 6.93% words are named-entities. As a result, a wrong `CRF` model which predicts every word as non-entity will result in high accuracy. Therefore, it is recommended to use F1-score instead of accuracy, F1-score considers both the precision (P) and the recall (R) of the test to compute the score, $F_1 = 2 * \frac{P*R}{P+R}$. If the model predicts the same thing for each input, the P or R will be zero, and F1 will be zero as well. Using F1-score solves the misguided score presented by accuracy.

3.2 Importance of Feature Extraction

Using the parameter settings explained in section 2.3, I will explain my experiences with feature extraction in this section. If we ignore features and only use the word itself as input, then the computed weighted F1-score is only 13.58%. After including common basic features (length of the word, isupper, islower, istitle, isdigit and isalnum), the weighted F1-score becomes 16.08%. However, the weighted F1-score jumps to 23.09% after adding the prefix and suffix up to ≤ 4 lengths. After adding isgazetteer, word-shape with short-word-shape, and pos-tag with chunk-tag, the computed weighted F1-score I get is 23.15%, 25.02%, and 25.13%, respectively. Adding features from neighbors' words increase the weighted F1-score to 27.29%. This clearly indicates that extracting features are important towards the improvement of named-entity

[3]https://sklearn-crfsuite.readthedocs.io/

[4]https://pypi.org/project/seqeval/

recognition system.

3.3 Parameters Optimization

Running the CRF model without any parameter results in low accuracy and weighted F1-score. Therefore, I feel the urge to play with the model's parameters during the training phase. I have focused on iterating over the different algorithms, max_iterations, c1 coefficient, and c2 coefficient.

Algorithm	Parameters	F1-Score
lbfgs	c1=0.9, c2=0.9	**0.26189**
l2sgd	c2=0.9	0.25206
ap	epsilon=1e-5	0.25390
pa	pa_type=1, c-0.9	**0.27139**
arow	variance=0.9, gamma=0.9	0.21059

Table 3: The weighted F1-score on algorithms.

For `algorithm`, we have five choices: 1) L-BFGS Gradient Descent (lbfgs), 2) L2 Stochastic Gradient Descent (l2sgd), 3) Averaged Perceptron (ap), 4) Passive Aggressive (pa), and 5) Adaptive Regularization Of Weight (arow). I have executed the CRF model on the different algorithms with default parameters. Table 3 shows that the `pa` algorithm performs 1% better than the `lbfgs` algorithm, however, the `lbfgs` algorithm has been using more frequently with the CRF model for a task like named-entity recognition.

For `max_iterations`, I have executed the CRF model with `verbose=True` for observing the F1-score after each iteration. I have run for 1000 iterations and find that the model reaches to the optimal F1-score around 100 epochs.

For `c1 and c2`, the range of coefficient value is 0 to 1, and I have iterated from 0 to 1 with a 0.1 step size. Figure 1 shows the change of weighted average F1-score over the [c1,c2] coefficient of the CRF model. For the 0 $\sim$ 1 range of [c1,c2], the F1-score fluctuates between 25% to 28%, which indicates the parameter tuning has impact on the model's score. The $c1 = 6.0000000000000035e-06$ and $c2 = 0.6000000000000001$ are computed as the best optimal value for Model-1 & Model-2 using the `RandomizedSearchCV` (sklearn crfsuite, v036) as a hyperparameter optimization technique, however, the most common value ($c1 = 0.1$ and $c2 = 0.1$) still gives the best result for Model-3.

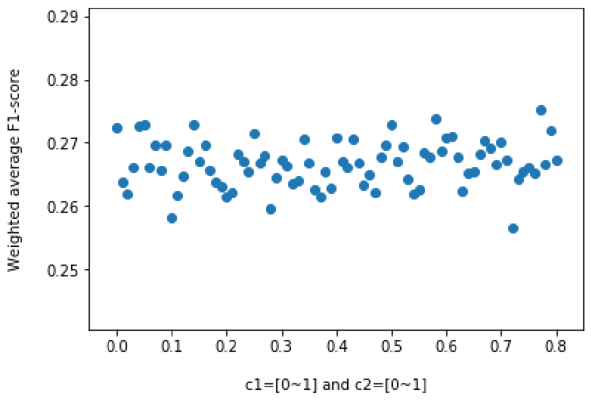
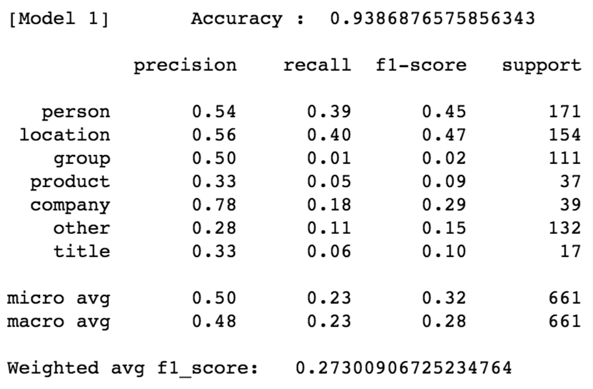


Figure 1: Change of the weighted F1-score over c1 & c2 coefficient of the CRF model.

3.4 Comparison of Models

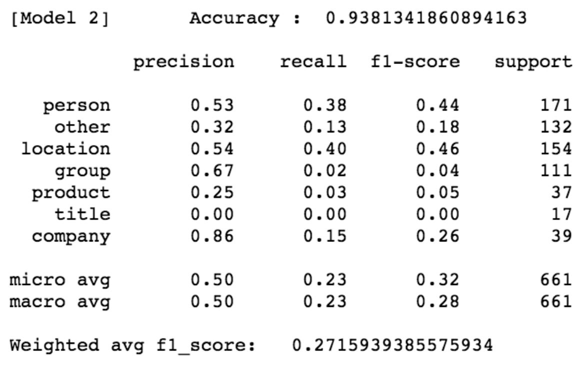
Table 2 shows the different sets of features that I have used to build different models. The accuracy, weighted average F1-score, and other metrics on the development set are shown in Figure 2- 4. Results are shown individually for each different category with micro, macro and weighted average F1-score.



```
[Model 1]        Accuracy :  0.9386876575856343

              precision    recall  f1-score   support

      person       0.54      0.39      0.45       171
    location       0.56      0.40      0.47       154
       group       0.50      0.01      0.02       111
     product       0.33      0.05      0.09        37
     company       0.78      0.18      0.29        39
       other       0.28      0.11      0.15       132
       title       0.33      0.06      0.10        17

   micro avg       0.50      0.23      0.32       661
   macro avg       0.48      0.23      0.28       661

Weighted avg f1_score:    0.27300906725234764
```

Figure 2: Result of Model-1 on the development set.

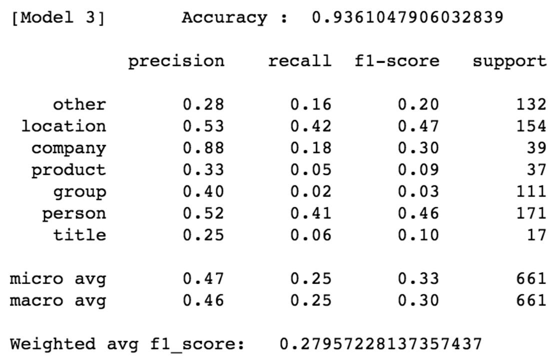
In Model-1 (Figure 2), I mainly focused on the feature list that the instructor discussed in class from the Figure 17.5 of Chapter 17.1 Named Entity Recognition (Daniel Jurafsky & James H. Martin, 2019). Here, the accuracy is 93.86% and the weighted average F1-score is 27.30% in the development set. However, the F1-score for 'group' entities is very low, 2% only.



```
[Model 2]       Accuracy :  0.9381341860894163

              precision    recall  f1-score   support

      person       0.53      0.38      0.44       171
       other       0.32      0.13      0.18       132
    location       0.54      0.40      0.46       154
       group       0.67      0.02      0.04       111
     product       0.25      0.03      0.05        37
       title       0.00      0.00      0.00        17
     company       0.86      0.15      0.26        39

   micro avg       0.50      0.23      0.32       661
   macro avg       0.50      0.23      0.28       661

Weighted avg f1_score:    0.2715939385575934
```

Figure 3: Result of Model-2 on the development set.

In Model-2 (Figure 3), I extended the feature list to improve the F1-score of 'group' entities, and the result shows that the F1-score of 'group' increased by 2% but highly penalized for 'title'.



```
[Model 3]       Accuracy :  0.9361047906032839

              precision    recall  f1-score   support

       other       0.28      0.16      0.20       132
    location       0.53      0.42      0.47       154
     company       0.88      0.18      0.30        39
     product       0.33      0.05      0.09        37
       group       0.40      0.02      0.03       111
      person       0.52      0.41      0.46       171
       title       0.25      0.06      0.10        17

   micro avg       0.47      0.25      0.33       661
   macro avg       0.46      0.25      0.30       661

Weighted avg f1_score:    0.27957228137357437
```

Figure 4: Result of Model-3 on the development set.

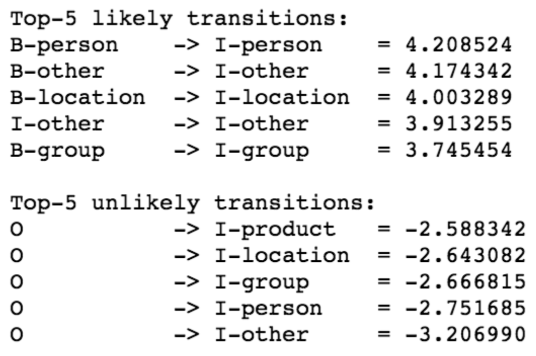
Therefore, I continued for several trails to build another Model-3 (Figure 4) with an optimized set of features that have 93.61% accuracy and 27.95% weighted average F1-score in the development set.

3.5 Top Transitions and States

In this section, I will show the top-5 likely/unlikely transitions and the top-5 positive/negative states. I have applied the `transition_features_` and `state_features_` options (sklearn crfsuite, v036) of the `CRF` model to compute the top transitions and top states, respectively.

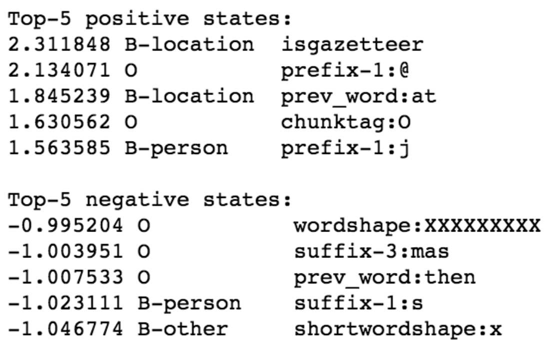
Figure 5 represents the top-5 likely transitions and the top-5 unlikely transitions in Model-1. Some observations are:

- It is very likely that the beginning of a named-entity (B-*) is followed by the inside of that named-entity (I-*).
- The transition from non-entity (O) to inside-entity (I-*) is quite unlikely.



```
Top-5 likely transitions:
B-person    -> I-person    = 4.208524
B-other     -> I-other     = 4.174342
B-location  -> I-location  = 4.003289
I-other     -> I-other     = 3.913255
B-group     -> I-group     = 3.745454

Top-5 unlikely transitions:
O           -> I-product   = -2.588342
O           -> I-location  = -2.643082
O           -> I-group     = -2.666815
O           -> I-person    = -2.751685
O           -> I-other     = -3.206990
```

Figure 5: Top-5 likely and unlikely transitions.



```
Top-5 positive states:
2.311848 B-location  isgazetteer
2.134071 O           prefix-1:@
1.845239 B-location  prev_word:at
1.630562 O           chunktag:O
1.563585 B-person    prefix-1:j

Top-5 negative states:
-0.995204 O           wordshape:XXXXXXXXX
-1.003951 O           suffix-3:mas
-1.007533 O           prev_word:then
-1.023111 B-person    suffix-1:s
-1.046774 B-other     shortwordshape:x
```

Figure 6: Top-5 positive and negative states.

Figure 6 represents the top-5 positive states and the top-5 negative states in Model-1. Some observations are:

- The model learns a word as a location entity if that word is a part of 'gazetteers' or the previous word is 'at'.
- Words starting with the '@' (i.e. user tag) are often not entities.
- Words having the previous word as 'then' or ends with 'mas' are likely entities of some kind.
- It is unlikely that person-entity ends with 's' or other-entity has a short word shape of 'x'.

4 Discussion

The named-entity recognition is mainly featuring an information extraction problem rather than the algorithm solving problem. The dataset we have given contains the raw text, however, we need an appropriate feature set in the target domain to train a model. It's not how many features we have that count, it's how many significant features we have. Among the extracted features, which subset of features is best suitable for the model is another challenging issue. Some important features can improve the named-entity recognition, on the other hand, irrelevant features can degrade the performance of the model. Therefore, finding the `significant` features is key to named-entity recognition.

Ambiguity also makes it much harder, as people usually do not follow the convention of writing. In addition, misspelling, punctuation, emoji, symbols, etc have been frequently used in sentences that increase the possibility of outliers for named-entity recognition.

The best reported weighted average F1-score is lower than 30%. There is plenty of room for improvement. At first, we can invest more processing power on parameter tuning to find the optimum value of each parameter to improve the accuracy of the model. A better understanding of the meaning of these parameters and their individual impacts on the model is essential to learn for tuning these parameters. We can train the model with more data or even can perform ensemble training. Some advance knowledge about model training (i.e. regularization, early-stop) may also help us to get rid of the bias-variance trade-off. Furthermore, we can explore more with a different set of feature selection until the score improves. Feature selection includes only informative features and removes all in-informative features.

Finally, processing the dataset is also important. Datasets may contain multiple blank lines or carriage returns. We have to carefully look at those issues, otherwise, the evaluation script may break with the dataset, and may raise unnoticeable consequence that may result in poor training and low score.

5 Conclusion

Recognizing named-entities is a challenging task and difficult to achieve the high F1-score for unstructured text. Implementing named-entity recognition is not only just building a model, but also to extract the more meaningful and important features to the task. The experiment with features and parameters emphasis that finding the best features and optimal parameter setting is key to improve the `NER` system.

References

Daniel Jurafsky & James H. Martin. Draft of August 29, 2019. *17.1 Named Entity Recognition, Speech and Language Processing*. `https://web.stanford.edu/~jurafsky/slp3/17.pdf`

Jacob Esienstein. Draft of November 13, 2018. *7.5.3 Conditional random fields, Natural Language Processing*. `https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf`

sklearn crfsuite. June 22, 2017 (v0.3.6). *The scikit-learn-compatible CRFsuite (python-crfsuite) wrapper*. `https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html`